*Proceedings of AFCEA's Sixteenth Annual Federal Database Colloquium and Exposition,*
"Information Dominance and Assurance" San Diego, CA, September 21-23, 1999.

# An Expanded Review of Information-System Terminology

**Marion G. Ceruti, Ph.D.,** Member, AFCEA
Engineering Group, Code D4121
Advanced Concepts and Systems Technology Division
Space and Naval Warfare Systems Center, San Diego
53560 Hull Street, San Diego, CA 92152-5001, USA
Tel. (619) 553-4068, Fax (619) 553-5136, ceruti@spawar.navy.mil

## Abstract

*Terminology pertaining to database systems is reviewed, particularly with respect to relational database systems; heterogeneous, distributed database systems; and information-management methods. The literature of a variety of database researchers and data administrators is included in this review. A comparison between the different ways in which some of the terminology is used in the industry is presented. In some cases, different definitions of the same term can be consistent when these definitions pertain to different aspects of the entity that the term represents. In other cases, popular misconceptions about word meanings are explained. Resolutions to conflicts in meaning and usage are suggested. Both the similarity and the diversity of ideas concerning the most basic, as well as the more complex database concepts are covered. For example, the discussions range from an examination of the word, data, in the general section, to the classification of synonyms in the section on semantic heterogeneity. This expanded review contains further comparison between types of data associations and between information storage and retrieval methods. It includes a comparison between knowledge bases and databases.*

Keywords - database system, information system, nomenclature, terminology

## 1. Introduction

The rapid growth of information-system technology during the last two decades has been accompanied by an equally rapid growth in the literature. These publications, technical manuals, and marketing brochures have originated from authors who exhibit a wide variety of training, background, and experience. Although this has resulted in an expansion of technical vocabulary, the growth of standards, particularly with regard to a comprehensive, uniformly accepted terminology, has not kept pace with the growth in the technology itself. Consequently, the nomenclature used to describe various aspects of information-system technology is characterized, in some cases, by confusion and chaos. The state of imprecision in the nomenclature of this field persists across virtually all data models and their implementations. The purpose of this paper is to highlight some areas of conflict and ambiguity and, in some cases, to suggest a more meaningful usage of the terminology. This work is a continuation of previous efforts to examine, describe, compare and evaluate the terminology used to describe information systems. (See, for example, [9,10,11 and 14]).

## 2. General information-system terms

### 2.1 What does the word, *data* mean?

According to Webster, the word, *data,* is a plural noun that refers to things known or assumed; facts or figures from which conclusions can be inferred; information. Derived from the Latin word, *datum,* meaning gift or present, data can be given, granted or admitted; premises upon which something can be argued or inferred. Although the word, *data*, is most frequently observed, the singular form, *datum*, is still used today, particularly as a reference for measuring tidal variations. More generally, a *datum* also is a real or assumed thing used as the basis for calculations [48].

The American National Standards Institute, Inc., (ANSI), the standards of which have been adopted by the Federal Government, defines *data* as "any representation of entities, relationships, or attributes such as characters or analog quantities to which meaning is or might be assigned" [1].

The Department of Defense defines *data* as a representation of facts, concepts, or instructions in a formalized manner suitable for communication, interpretation, or processing by humans or by automatic means [38]. The word, *data*, also is an adjective in terms such as *data set*; *data fill*; *data resource*; *data management*; and *data mining*. A *data set* is an aggregate of related data items.

Implicit in all definition of *data* is the notion that the user can reasonably expect data to be true and accurate. For example, a data set is assumed to consist of facts that are given for use in a calculation or an argument, for drawing a conclusion, or as instructions from a superior authority. This also implies that the data-management community has a responsibility to ensure the accuracy, consistency, and currency of data.

## 2.2 Data element versus data item

ANSI's definition of *data element* as "(1) a named unit of data. It can be used to describe the atomic level of data, whether computerized or manual, as viewed by the user. (2) in database usage, a named identifier of the entities and attributes that are represented in a database." [1]. This definition includes both the conceptual and the practical views.

In an attempt to define database terms with a view toward practical applications, the Department of Defense (DOD) defines a *data element* as a named identifier of each of the entities and their attributes that are represented in a database [38]. As such, *data elements* must be designed:

- to represent the attributes (characteristics) of data entities identified in data models;
- according to functional requirements and logical (as opposed to physical) characteristics;
- according to the purpose or function of the *data element*, rather than how, when, where, and by whom it is used;
- with singularity of purpose such that it has only one meaning; and
- with well-defined, unambiguous, and separate domains [38].

Glymph defines a *data element* as data described at the useful primitive level [27].

Fortier et al. define *data item* as the smallest separable unit recognized by the database representing a real-world entity [24].

What is clear from the above definitions is that there is considerable ambiguity in what these terms mean. The author proposes the following distinction between *data element* and *data item*: A *data element* is a variable associated with a domain (in the relational model) or an object class (in the object-oriented model) characterized by the property of atomicity. A *data element* represents the smallest unit of information at the finest level of granularity present in the database. An instance of this variable is a *data item*, or a *datum.* A *data element* in the relational model is simply an attribute (or column) that is filled by data items commonly called the "data fill." This distinction clarifies but does not preclude any definition given above.

## 2.3 Definitions of data model

*Data model* is a term that has been used in two related but different ways. Its definitions are as follows: 1. a very general category of data-management methodology, such as the relational model or the object-oriented *data model*, and 2. a diagramatic or otherwise systematic representation of the information that an enterprise uses to show the relationships between data elements, data sets, etc. In this sense, the *data model* captures information about the data structures that support business-area requirements. ANSI's definition, which also emphasizes that data models pertain to the enterprise, is "a description of the organization of data in a manner that reflects the information structure of the enterprise" [1]. ANSI also relates this to the notion of a data structure, which it defines in two ways. First, it is a set of logical relationships that exist between units of data, and second, the term is used to describe an instance or occurrence of a data model [1]. Examples of *data models* include the Command and Control (C2) Core

Data Model [21], the IRDS Data Model [2], and the DoD Enterprise Data Model. Different types of models apply to different levels of architecture in an information system [31].

## 2.4 What is a database?

The many definitions for the term, *database*, range from the theoretical and general, to the implementation specific, depending on one's point of view. ANSI distinguishes *database* (one word) from the closely associated term, *data base* (two words) [1]. ANSI lists multiple definitions for both terms. A *database* (DB) is (1) a large collection of interrelated data stored together to serve one or more applications; "(2) in CODASYL, the data defined and described by one schema; (3) a repository for data sufficient to serve some purpose of an enterprise on a continuing basis, and (4) See also *data base* [1]." A *data base*, according to ANSI, has two definitions: "(1) a set of data, part or the whole of another set of data, consisting of at least one file, that is sufficient for a given purpose or for a given data processing system; (2) See also database. [1]"

For example, Brathwaite, Darwen and Date have offered two different, but not necessarily inconsistent definitions of *database* that are specific to the relational model [3 and 20]. Darwen and Date have built their definition on fundamental constructs of the relational model and is very specific to that model. Brathwaite has employed a definition that is based on how *databases* are constructed in a specific Database Management System (DBMS). These definitions are discussed in section 3.

Actually the word, *database*, can have multiple definitions depending on the level of abstraction under consideration. For example, Sheth and Larson define *database* in terms of a reference architecture, in which a *database* is a repository of data structured according to a data model [44]. This definition is more general than that of either Brathwaite or Darwen and Date because it is independent of any specific data model or DBMS. It could apply to hierarchical and object-oriented *databases* as well as to relational *databases*; however, it is not as rigorous as Darwen and Date's definition of a relational *database* because the term, repository, is not defined.

Similarly, Fortier et al. defines a *database* to be a collection of data items that have constraints, relationships and a schema [24]. Of all the definitions for database considered thus far, the one in [24] is the most similar to that of Sheth and Larson [44], because the term *data model* could imply the existence of constraints, relationships and a schema. Moreover, Fortier et al. define *schema* as a description of how data, relationships and constraints are organized for user-application program access [24]. A *constraint* is a predicate that defines all correct states of the database [24]. Implicit in the definition of *schema* is the idea that different schemata could exist for different user applications. This notion is consistent with the concept of multiple schemata in a Federated Database System (FDBS). Terms germane to FDBSs are discussed in a subsection 5.2.

Waldron defines *database* as a collection of interrelated files stored together where specific data items can be retrieved for various applications [47]. A *file* is a collection of related records [47]. Similarly, Wheeler defines *database* as a collection of data arranged in groups for access and storage; a *database* consists of data, memo and index files [49].

According to Levesque and Brachman, a *database* is a *knowledge base* with a limited form that permits a very special form of inference [37]. That limited form contributes to the tractability of the knowledge-representation service in the *database* by limiting its expressiveness, particularly with regard to expressing some kinds of uncertainty [37]. In the relational model, one special form of inference is implemented through the use of relational calculus in Relational Database Management Systems (RDBMSs), the tractability being partially responsible for the popularity of RDBMSs. Most of the time, however, the terms, *database* and *knowledge base* are used to denote different forms of information representations. (See section 3 on relational databases and subsection 2.7 on knowledge bases.)

To summarize this subsection, it is evident that all definitions of *database* either explicitly or implicitly include the following concepts:

- Data in *databases* can be arranged according to a schema, or a data structure.
- Relationships exist between data in a *database* that influence the structure.
- Depending on the kind of *database* in question, specific methods and techniques are available to access of the stored data.

- Data are stored in one place, either at the same physical location, or in a distributed manner in which users can issue queries to aggregate the data from different sources on the same client platform. (See subsection 5.1 on distributed databases.)

## 2.5 Data repository and database system

The terms, *data repository* and *database system* are very closely related and not mutually exclusive. Both terms refer to a more comprehensive environment than described by most definitions of the term, *database*, because they are concerned with the means necessary for the management of data in addition to the data themselves and their structures. The ANSI definition number three of *database* captures the concept [1] of a *data repository.* It implies that a database can be a repository of data if it is sufficient to serve some purpose of an enterprise on a continuing basis. This, in turn, implies the presence of tools, procedures, personnel and life-cycle support to meet the requirements of serving the enterprise on a long-term basis, as opposed to an ad hoc database constructed for a short-term project. This is consistent with the notion that a *data repository* is the heart of a comprehensive information management system environment [3].

For example, it must include not only data elements but metadata that are of interest to the enterprise, data screens, reports, programs, and systems [3]. A *data repository* must provide a set of standard entities and allow for the creation of new, unique entities of interest to the organization [3]. King et al. describe characteristics of a *data repository* to include in internal set of software tools, a DBMS, a meta model, populated metadata, and loading and retrieval software for accessing repository data [35].

Similarly, a *database system* (DBS) includes both the DBMS software and one or more databases [44]. A database system also can be a *data repository* that can include a single database, or several databases.

## 2.6 What are information and knowledge?

Webster's defines *information* in many ways. A few definitions of interest to information-systems engineers are as follows [48]:

- The communication or reception of knowledge or intelligence,
- Knowledge obtained from investigation, study or instruction,
- Intelligence, news, facts, data,
- A signal or character, as in a communication system or computer, representing data.

As these definitions emphasize the communication or transmission of aspect of *information,* if facts or data cannot be communicated, their value as *information* is diminished to the point of uselessness.

Of the many definitions found in Webster's dictionary for the word, *knowledge*, the definitions of interest to information-system researchers, engineers and developers are as follows [48]:

- The range of one's information or understanding,
- The fact or condition of apprehending truth or fact,
- The fact or condition of having information,
- The sum of what is known: the body of truth, information, and principles acquired by mankind.

According to Goodyear, et al. a definition of *knowledge* on which everyone can agree may not be possible to find [25]. However, for the purpose of discussion, they define *knowledge* as "complex content" that, in aggregate, has attained a level of complexity beyond that of traditional transactional data, to include figures, text, voice, images, video and other media [25]. To be *knowledge*, the content must be capable of capture, storage and delivery to users [25].

Together, these definitions imply that *knowledge* results from an aggregate of facts or data on which some logical reasoning, analysis, computation, and/or correlation has been performed to interpret or summarize the information implicit in the data and to raise the level of understanding of the observers who will use the *knowledge*. The notion of complexity is a key distinction between data and *knowledge.*

The complexity inherent in *knowledge* often is expressed as declarative statements, rules, principles, or statements of truth, such as those concerning the relationships between an arbitrary quantity of observable data. The level of complexity and expressiveness that can be achieved by aggregating data into information and knowledge is limited only by the capabilities of our information systems and our ability to manage them.

## 2.7 What is the difference between a database and a knowledge base?

As is the case for the term, *knowledge* a comprehensive, exclusive, and unique definition of the term, *knowledge base* that will satisfy all knowledge engineers is not available [14]. However, it is still useful to attempt to define *knowledge base* in a meaningful manner that includes all cases that pertain to the use of *knowledge bases* with inference engines in the field of artificial intelligence (AI), and that excludes cases that are not intended to be part of the an AI scenario. Considering the diversity in various definitions that have been proposed, the information-management community needs a better definition of *knowledge base* [14]. Thus, a working definition that will cover many cases is as follows: "Unlike a database that stores information implicitly in tabular format, a *knowledge base* is a source of information that stores facts explicitly as declarative assertions, sometimes in the form of frames and slots, sometimes in the form of probabilistic networks, and sometimes in other forms that facilitate reasoning with inference engines." This open-ended definition seems like an oxymoron; however, some flexibility is necessary because evolving AI technology continues to makes more and more sources of information accessible to reasoning software, machine learning and automated knowledge acquisition. (See, for example, [10 and 14].)

For example, a number of web-accessed document-management "databases," some of which are called *knowledge bases*, can be queried for specific document segments, [14]. These actually are text files rather than declarative assertions formatted especially for input into inference engines [14]. The information contained in them is written as plain text in a common language, such as in English, French, or some other non-programming language. These textual information sources meet the requirements of the definition of *knowledge base* because information is stored declaratively, and text-based knowledge discovery can be performed. The test of "Can the candidate *knowledge base* be input into an inference engine?" fails as a means to exclude web-based text documents because some web browser software designed for AI purposes can read text, access web sites, and extract keywords, etc. for expert systems to use [14].

One of the most important observations regarding the differences between databases and *knowledge bases* is the degrees of expressiveness and tractability [37]. *Knowledge bases* tend to be much more expressive (and less tractable) than databases, whereas databases are more tractable but sacrifice some expressiveness to gain that tractability [37]. A *knowledge base* can deal with uncertainty much more expressively than a database can. For example, when you encounter a null value in a database, do you always know whether is it because the information is unknown, not applicable, unavailable at the time of the last update, or classified at a higher level?

## 2.8 Data warehouses, data marts and data stores

Thuraisingham [45 and 46] and Wysong [51] have discussed the importance of the *data warehouse*, which is a database system that is optimized for the storage of aggregated and summarized data across the entire range of operational- and tactical-enterprise activities [51]. According to *data-warehouse* pioneer, W. Inmon, a *data warehouse* is a subject-oriented, integrated, nonvolatile, and time-variant collection of data to support management's strategic decision-making process for the enterprise [30 and 31]. The data warehouse brings together several, possibly heterogeneous, databases from diverse sources in the same environment [45]. For example, this aggregation could include data from current systems, legacy sources, historical archives, and other external sources [51].

Unlike databases that are optimized for rapid retrieval of information during real-time transaction processing for tactical purposes, *data warehouses* are not updated, nor is information deleted. Rather, time-stamped versions of various data sets are stored. *Data warehouses* also contain information such as summary reports and data aggregates tailored for use by specific applications [45]. Thus, the role of metadata is of critical importance in the extracting, mapping, and processing data to be included in the warehouse [51]. All of this serves to simplify queries for the users, who query the *data warehouse* in a read-only, integrated environment.

Table 1. Comparison between different information storage and retrieval methods

| Data-Storage Method | Time Variance | Time Horizon | Updates (Volatile) | Integrated | Subject Oriented | Knowledge |
|---|---|---|---|---|---|---|
| Database | No - Current values of data are stored. | Several Months | Yes, dynamic | Maybe (and maybe NOT) | Most | Implicit |
| Data Warehouse & Data Mart | Yes - historic, "Snapshots" of data | 5-10 years or more | No, static | Yes | Yes | Implicit |
| Operational Data Store | Current data values | Several Months | Yes, dynamic | Yes | Yes | Implicit |
| Knowledge Base | Maybe | Varies | Maybe | Maybe | Most | Declarative, and explicit |

A *data mart* contains data from a *data warehouse* tailored for specific analytical requirements of a given business unit or function [30]. A *data mart* is designed and constructed to support a specific set of users. This is why Table 1 shows the same characteristics for both a *data mart* and a *data warehouse.*

A *data store* is a subject-oriented, integrated, volatile, and current collection of data to support management's tactical decision-making process for the enterprise [32]. Table 1 enables a comparison between this definition and that given above for a *data warehouse*.

## 2.9 Data mining and knowledge discovery

The *data warehouse* is designed to facilitate the strategic, analytical, and decision-support functions within a organization. One such function is *data mining*, which is the search for previously unknown information in a *data warehouse* or database containing large quantities of data [45]. Similarly, Goodyear, et al. have defined *data mining* as "the process of extracting valid and understandable but previously unknown information from large data stores in a format that supports making useful business decisions" [26]. The *data warehouse* or database is analogous to a mine, and the information desired is analogous to a mineral or precious metal. The concept of *data mining* implies that the *data warehouse* or *data store* in which the search takes place contains a large quantity of unrelated data and probably was not designed to store and support efficient access to the information desired. In *data mining*, users expect that multiple, well-designed queries and a certain amount of data analysis and processing will be necessary to summarize and present the information in an acceptable format.

The *data warehouse* also is an appropriate starting point for *knowledge discovery*, which is a form of *data mining* that can use various techniques such as artificial intelligence, machine learning, and neural networks to discover new relationships between variables in large quantities of data and information [42]. (See also, subsection 2.6 on *knowledge.*)

## 2.10 Data administrator versus database administrator

The following discussion is not intended to offer an exhaustive list of tasks performed by either *Data Administrators* (DA) or *Database Administrators* (DBA), but rather to highlight what have traditionally been the similarities and essential distinctions between these two types of database professionals. As one might expect, both DAs and DBAs are concerned with the management of data, but at different levels.

ANSI defines *data administration* as "the responsibility for definition, organization, supervision, and protection of data" [1]. It is related to the idea of data resource management [1]. Thus, the job of a *data administrator* is to set policy about determining the data an organization requires to support the processes of that organization. The DA develops or uses a data model and selects the data sets that the database will support. A DA collects, stores, and disseminates data as a globally administered and standardized resource [3]. Data standards on all levels that affect the organization fall under the purview of the DA, who is truly an administrator in the managerial sense. For a more comprehensive discussion of *data administration*, see [3].

By contrast, a *database administrator's* technical orientation is at a finer level of granularity than that of a DA. For this reason, in very large organizations, DBAs focus on only a subset of the organization's users. Typically, the DBA, like a computer systems manager, is charged with day-to-day, hands-on use of the DBS and daily interaction with its users. DBAs are familiar with the details of implementing and tuning a specific DBMS or a group of DBMSs. For example, the DBA has the task of creating new user accounts, programming the software to implement a set of access controls, and utilizing audit functions.

To illustrate the distinction between a DA and a DBA, the U.S. Navy has a head *data administrator*, whose range of authority extends throughout the entire Navy. It would not be practical or possible for an organization as large as the Navy to have a DBA in an analogous role, because of the multiplicity of DBSs and DBMSs that are in use and the functions that DBAs perform.

These conceptual differences not withstanding, in smaller organizations, a single individual can act as both DA and DBA, thus blurring the distinction between these two roles.

Moreover, because data models and standards have increased in complexity, DAs now rely increasingly on new technology to accomplish their tasks, just as DBAs do.

## 3. Relational database terms

Because relational technology is a mature technology with a rigorous relational algebra and many practical applications, some of the important terms that pertain to the relational model are described here. Many of these terms are straightforward and generally unambiguous, whereas some terms have specific definitions that are not always understood. A data set that is represented in the form of a *table* containing *columns* and *rows* is called a *relation.* The columns are called *attributes*, and the rows are called *tuples* [3]. Darwen and Date define a tuple to be a set of ordered triples of the form <A, V, *v*> where A is the name of an attribute, V is the name of a unique *domain* that corresponds to A, and *v* is a value from domain V called the *attribute value* for attribute A within the tuple [20]. A *domain* is a named set of values [20]. Darwen and Date also describe a relation as consisting of a *heading* and a *body*, where the heading is a set of ordered pairs, <A,V>, and the body consists of tuples, all having the same heading <A,V> [20]. N.B. An attribute value is a data item or a datum.

In some respects, a relation is analogous to an array of data created outside of an RDBMS, such as in a third-generation language (3GL) program like C, FORTRAN or Ada, in which the rows are called *records* and the columns are called *fields*. Waldron defines a *field* as a set of related letters, numbers or other special characters, and a *record* as a collection of related fields [47].

Some major RDBMS vendors have underscored the interchangeability of the terms *record* and *row* by the way in which they report the results of a query to the user. Earlier versions of commercial DBMSs indicated at the end of a query return, screen messages of the form "12 records selected." In later versions of the DBMS, it is more common to see "12 rows selected" or "12 rows affected" instead.

### 3.1 Relation versus relation variable

Darwen and Date have called attention to the correct manner in which the term *relation* should be used [18, 19 and 20]. The definition given above specifically includes values, *v*, from domain, V. However, the term, *relation*, has not always been used correctly in the industry. "Relation" frequently is used as though it could mean either a filled table with data present (correct), or an empty table structure containing only data headers (incorrect.) The confusion here is due to a failure to distinguish between a *relation*, which is a filled table with tuples containing attribute values, and *relation variable* (or *relvar*), which is an empty table structure with only attribute names and domains from which to choose values. The values of a *relation variable* are the *relations* per se [19]. This distinction becomes particularly important when mapping between the relational and object-oriented data models [8 and 19].

### 3.2 Database versus database variable

In a manner similar to the relation-relvar dichotomy, a *database variable* is different from a *database* per se. A *database variable* (or *dbvar*) is a named set of relvars [20]. The value of a given *dbvar* is a set of specific, ordered pairs <R,r>, where R is a relvar and r (a relation) is the current value of that relvar, such that one such ordered pair exists for each relvar in the *dbvar*, and that taken together, all relvar values satisfy the applicable constraints (in particular, integrity constraints.) A value of the dbvar that conforms to this definition is called a *database* [20]. Some call this a *database state*, but this term is not used very often.

### 3.3 Database versus DBMS

As can be seen from the above examples, not all information-system terminology is as unambiguous as "rows" and "columns." Incorrect understanding of the fundamental concepts in database technology can lead to inconsistent terminology, and vice versa. For example, *databases* frequently are described according to the *DBMS* that manages them. This is all well and good as long as one realizes that phrases like "Oracle database" and "Sybase database," refer to the *databases* that are managed using Oracle or Sybase software. Difficulty arises when this nomenclature results in the misconception that *DBMS* software is actually the *database* itself. The assumption that Informix, for example, is a *database* is as illogical as thinking that the glass is the same as the water in it.

### 3.4 Concept versus implementation in relational databases

Darwen and Date's definition of a *database* [20], as well as those of other database researchers, (some of whom are mentioned in the references of this chapter and others who have not), do not require the presence of a DBMS. Conceptually, it is possible to have a database without a DBMS or a DBMS without a database, although obviously, the greatest utility is achieved by combining the two. In the context of a specific DBMS environment, Brathwaite defines an IBM Database 2 (DB2) database as a collection of table and index spaces where each table space *can* contain one or more physical tables [3]. This definition is inconsistent with Date's definition because it allows for the possibility that the table spaces could be empty, in which case no data would be present. It is not clear that even relvars would be present in this case. That not withstanding, if physical tables are present, Brathwaite's definition becomes an implementation-specific special case of Date's definition. (Substitute the word *must* for *can* to resolve the conflict with Brathwaite's definition.)

N.B. Except in the case where the vendor has specified default table and index spaces in the DBMS code, the database and index spaces are not actually part of the DBMS per se. The DBA needs to create both the database space and the index space using the DBMS software.

## 4. Database normalization

The topic of *database normalization*, sometimes called *data normalization* has received a great deal of attention in the literature [4, 16, 21, 33 and 50]. As is usually the case, database normalization is discussed below using examples from the relational data model. Here, the terms *relation* and *table* will be used interchangeably. However, the design guidelines that pertain to database normalization are useful even if a relational DBS is not used [34]. For example, Lee has discussed the need for normalization in the object-oriented data model [36].

### 4.1 What is database normalization?

Strictly speaking, *database normalization* is the arrangement of data into tables. The terms, *data normalization* and *database normalization,* have been used incorrectly to refer to processes that are better described as data standardization and/or database integration. Whereas *normalization* is part of the data modeling processes for data-element standardization [21], it does not constitute this process in its entirety.

Winsberg [50] defines normalization as the process of structuring data into a tabular format, (with the implicit assumption that the result must be at least in first normal form.) Similarly, Brathwaite [4] defines *data normalization* as a set of rules and techniques concerned with:

- Identifying relationships between attributes
- Combining attributes to form relations (with data fill)
- Combining relations to form a database

The chief advantage of *database* or *data normalization* is to avoid modification anomalies [4] that occur when facts about attributes are lost during insert, update and delete operations. However, if the normalization process has not progressed beyond first normal form, it is not possible to insure that these anomalies can be avoided. Therefore, *database normalization* commonly refers to further *nonloss decomposition* of the tables into second through fifth normal form [50]. *Nonloss decomposition* means that information is not lost when a table in a lower normal form is divided (according to attributes) into tables that result in the achievement of a higher normal form. This is accomplished by the placing primary and foreign keys into the resulting tables, so that tables can be joined to retrieve the original information.

## 4.2 What are normal forms?

A *normal form* of a table or database is an arrangement or grouping of data that meets specific requirements of logical design, key structure, modification integrity, and redundancy avoidance, according to the rigorous definition of the normalization level in question. A table is said to be in "X" normal form if it is already in "X-1" normal form and it meets the additional constraints that pertain to level "X."

In *first normal form* (1NF), related attributes are organized into separate tables, each with a *primary key*. A *primary key* is an attribute or set of attributes that uniquely define a tuple. Thus, if a table is in 1NF, entities within the data model contain no attributes that repeat as groups [21]. Kent has explained that in 1NF, all occurrences of a record must contain the same number of fields [34]. In 1NF, each data cell (defined by a specific tuple and attribute) in the table will contain only atomic values.

Every table that is in *second normal form* (2NF) must be in 1NF and also must meet the condition that every non-key attribute depends on the entire primary key. Any attributes that do not depend on the entire key are removed and placed in a separate table to preserve the information that they represent. 2NF becomes an issue only for tables with *composite keys* [34]. A *composite key* is defined as any key (candidate, primary, alternate or foreign) that consists of two or more attributes [50]. If only part of the *composite key* is sufficient to determine the value of a non-key attribute, the table is not in 2NF.

Every relation that is in *third normal form* (3NF) must be in 2NF and also every non-key attribute depends directly on the entire primary key [50]. In 2NF, non-key attributes are allowed to depend on each other. This is not allowed in 3NF. No non-key attribute within an entity determines the value of another non-key attribute in 3NF [21]. If an attribute does not depend on the key directly, or if it depends on another non-key attribute, it is removed and placed into a new table. It is often stated that in 3NF, every non-key attribute is a function of the key, the whole key, and nothing but the key [4 and 50]. In 3NF, every non-key attribute must contribute to the description of the key. However, 3NF does not prevent part of the primary key from depending on a non-key attribute, nor does it address the issue of candidate keys.

*Boyce-Codd normal form* (BCNF) is a stronger version of 3NF. Every relation that is in BCNF must be in 3NF and must meet the additional requirement that each *determinant* in the table must be a candidate key. A *determinant* is any attribute of a table that contains unique data values, such that the value of another attribute fully functionally depends on it. A *candidate key* is any attribute or group of attributes, including the primary key, that is sufficient to define the tuple uniquely and that can act as the key to the relation. If a *candidate key* also is a composite key, each attribute in the composite key must be necessary and sufficient for uniqueness. Winsberg calls this condition "unique and minimal" [50]. Primary keys meet these requirements. An *alternate key* is any *candidate key* that is not the primary key. In BCNF, no part of the key is allowed to depend on any non-key attribute. Compliance with the rules of BCNF forces the database designer to store associations between determinants in a separate table, if these determinants do not qualify as candidate keys.

BCNF removes all redundancy due to singular relationships but not redundancy due to many-to-many relationships [50]. To do this, further normalization is required. *Fourth* and *fifth normal forms* (4NF and

5NF) involve the notions of multivalued dependence and cyclic dependence, respectively. A table is in 4NF if it is in BCNF and it also does not contain any independent many-to-many relationships.

That not withstanding, a table could be in 4NF and still contain dependent many-to-many relationships. A table is in 5NF if it is in 4NF and it also does not contain any cyclic dependence (except for the trivial one between candidate keys [50].) In theory, 5NF is necessary to preclude certain join anomalies, such as the introduction of a false tuple [50]. However, in practice the large majority of tables in operational databases do not contain attributes with cyclic dependence.

### 4.3 What are over normalization and denormalization?

*Over normalization* of a table results from further nonloss decomposition that exceeds the requirements to achieve 5NF [50]. The purpose of this is to improve update performance. However, most operational databases rarely reach a state in which the structure of all tables has been tested according to 5NF criteria, so *over normalization* rarely occurs. *Over normalization* is the opposite of *denormalization*, which is the result of intentionally introducing redundancy into a database design to improve retrieval performance. In the case of denormalization, the database-design process may have progressed to 3NF, BCNF, 4NF, or even to 5NF. However, the database is implemented in a lower normal form to avoid time-consuming joins. Because the efficiency of "select" queries is an issue in operational systems, *denormalization* is more common than *over normalization*.

The first six normal forms (including BCNF) are formal structures of tables that eliminate certain kinds of intra-table redundancy. For example, 5NF eliminates all redundancy that can be removed by dividing tables according to attributes. Higher normal forms exist beyond 5NF. They address theoretical issues (such as dividing tables according to tuples instead of attributes) that are not considered to be of much practical importance [50]. In fact, Date [16] has noted that often it is not necessary or desirable to carry out the normalization process too far because normalization optimizes update performance at the expense of retrieval performance. Most of the time, 3NF is sufficient. This is because tables that have been designed logically and correctly in 3NF are almost automatically in 4NF [16]. Thus, for most databases that support real-time operations, especially for those that have tables with predominantly single-attribute primary keys, 3NF is the practical limit. Note that a two-attribute relation with a single-attribute key is automatically in the higher normal forms.

## 5. Nomenclature of distributed, heterogeneous and combined data systems

### 5.1 What is a distributed database?

Date defines a *distributed database* as a virtual database that has components physically stored in a number of distinct "real" databases at a number of distinct sites [17].

### 5.2 Federated database systems versus multidatabase systems

Hammer and McLeod coined the term, *Federated Database System* (FDBS) to mean a collection of independent, pre-existing databases (for which the data administrators and/or the database administrators) agreed to cooperate [28 and 29]. Thus, the DBA for each component database provides the federation a schema that represents the data from his or her component that can be shared with other members of the federation [28].

In a landmark paper [44], Sheth and Larson define FDBS in a similar but broader architectural sense to mean a collection of cooperating but autonomous component database systems that are possibly heterogeneous. They also define a *nonfederated database system* as an integration of component DBMSs that are not autonomous with only one level of management, in which local and global users are not distinguished [44]. According to the taxonomy presented in [44], both federated and nonfederated database systems are included in a more general category called *multidatabase systems*. These multidatabase systems support operations on multiple component DBSs.

Sheth and Larson further divide the subcategory of FDBS into two types: *loosely coupled* and *tightly coupled* FDBS based on who creates and maintains the federation and how the component databases are integrated. If the users themselves manage the federation, they call it a *loosely coupled* FDBS, whereas if a global DBA manages the federation and controls access to the component databases, the FDBS is *tightly coupled* [44]. Both loosely coupled and tightly coupled FDBSs can support multiple federated schemata. However, if a tightly coupled FDBS is characterized by the presence of only one federated schema, it has a *single federation* [44].

The term, *multidatabase*, has been used by different authors to refer to different things. For example, Litwin et al. [39] have used it to mean what Sheth and Larson call a loosely coupled FDBS. By contrast, Breitbart and Silberschatz [5] have defined multidatabase to be the tightly coupled FDBS of Sheth and Larson. Reference [44] contains additional examples of the variety of conflicting ways in which the term, *multidatabase,* has been used.

The above examples by no means constitute the only definitions for the terms, *loosely coupled FDBS* and *tightly coupled FDBS,* that occur in the literature. While otherwise accepting the general structure of Sheth and Larson's taxonomy [44], Ceruti and Kamel have used the terms, *loosely coupled* and *tightly coupled FDBS* to distinguish between the degree to which users can perceive heterogeneity in an FDBS, among other factors [7 and 13]. In this system of nomenclature, a *tightly coupled* FDBS is characterized by the presence of a federated or global schema, which is not present in a *loosely coupled* FDBS. Instead of a global schema, *loosely coupled* FDBSs are integrated using other software, such as a user interface with a uniform "look and feel," or a standard set of queries used throughout the federation, thus contributing to a common operating environment. In this case, the autonomous components of a *loosely coupled* FDBS are still cooperating to share data, but without a global schema. Thus, the users see only one DBS in a *tightly coupled* FDBS, whereas they are aware of multiple DBSs in the *loosely coupled* FDBS. Here, *tightly coupled* FDBS obey Date's rule zero, which states that to a user, a distributed system should look exactly like a nondistributed system [23].

Given this manner in which to characterized an FDBS [7], a *hybrid* FDBS is possible for which some of the component DBSs have a global schema that describes the data shared among them (*tightly coupled*), but other components do not participate in the global schema (*loosely coupled*).

The author proposes that the taxonomy of Sheth and Larson be combined with that of Ceruti and Kamel to provide a more comprehensive system to describe how databases are integrated. This expanded taxonomy accounts for the perspectives of both the DA and the users. Essentially, most aspects of Sheth and Larson's taxonomy are logical and should be retained. However, instead of using Sheth and Larson's terms for *tightly coupled* federated database and *loosely coupled* federated database, the terms *tightly controlled* federated database and *loosely controlled* federated database, respectively should be substituted. This change will focus on the absence or presence of a central controlling authority as the essential distinction between the two. In this new usage, the terms, *tightly coupled* and *loosely coupled* describe how the user, rather than the DA, sees the federation. Given this change, the *coupling* between components in a federated database will describe how seamless and homogeneous the database looks to users and applications.

The expanded taxonomy described above can accommodate federated databases that differ widely in their characteristics. For example, if a *tightly controlled federated database* is *tightly coupled*, the global data administrator and the global database administrator have exercised their authority and expertise to provide a seamless, interoperable environment that enables the federation's users to experience the illusion of a single database for their applications and ad hoc queries. A *tightly controlled federated database* can also be *loosely coupled*, in which case, the global data administrator allows the users of the federation to see some heterogeneity with respect to the component databases. Both conditions are within the realm of possibility. However, a *loosely controlled federated database* is almost certain to be *loosely coupled*. This is because a *loosely controlled federated database* lacks a central authority to mediate disputes about data representation in the *federated schema*, and to enforce uniformity in the federation's interfaces to user applications. A *loosely controlled federated database* is not likely to be *tightly coupled.*

## 5.3 Local or localized schema, component schema and export schema

A local or localized database generally starts as a stand-alone, non-integrated database. When a local, autonomous database is selected for membership in a federation, a *local schema* is defined as a conceptual schema of a the component DBS that is expressed in the native data model of the component DBMS [44]. When the local database actually becomes a member of a federated database, it is said to be a *component database*. The schema associated with a given database component is called a *component schema*, which is derived by translating a local schema into the common data model of the FDBS [44]. An *export schema* represents the subset of the *component schema* that can be shared with the federation and its users [44]. Similarly, Date defines a *local schema* as the database definition of a component database in a distributed database [17].

## 5.4 Federated schema, global schema and global data dictionary

A *federated schema* is an integration of multiple export schemata [44]. Because the distributed database definition sometimes is called the *global schema* [17], *federated schema* and *global schema* are used interchangeably. Additional synonyms for *federated schema* can be found in [44].

A *global data dictionary* is the same as a *global schema* that includes the data-element definitions as they are used in the FDBS. A *data dictionary* is different from a schema, or database structure specification, because a *data dictionary* contains the definitions of attributes or objects, not just the configuration of tables, attributes, objects, and/or entities within that structure. It is especially important to include the data-element definitions with the export schemata when forming a federated database, in which multiple data representations are likely. Simply having a collection of database structures is insufficient to complete a useful federated schema. It is necessary to know the meaning of each attribute or object and how it is construed in the component database.

## 5.5 Middleware versus midware

In a three-tiered, client-server architecture designed to connect and to manage data exchange between user applications and a variety of data servers, the middle tier that brokers transactions between clients and servers consists of *middleware*, which is sometimes called *midware*. Cykana defines *middleware* as a variety of products and techniques that are used to connect users to data resources [15]. In his view, the middleware solution usually is devoted to locating and finding data rather than to moving data to migration environments [15].

Cykana describes two options for middleware, depending on the degree of coupling between the user and the data resource [15]. *Loosely coupled middleware* products allow flexibility in specifying relationships and mappings between data items, whereas *tightly coupled middleware* products allocate more authority to standard interfaces and database administrators. (N.B. The term data item was not defined in [15].) Each option has its advantages and disadvantages. The *loosely coupled middleware* does not require the migration or legacy data structures to be modified, and it allows users to access multiple equivalent migration systems transparently with one standard interface [15]. The disadvantage of this option is that it does not prevent multiple semantics and non-standard structures [15].

*Tightly coupled middleware* is a more aggressive strategy that combines Applications Program Interface (API) and Graphic-User Interface (GUI) technologies, data communications, and data-dictionary design and development capabilities to provide distributed data access [15]. With this option, data standardization and reengineering are required [5].

The concept of loose and tight coupling to *middleware* described in [15] bears some similarity to, but also differs in some ways from, the loose and tight coupling between data resources discussed by Sheth and Larson [44] and other researchers. In the case of *middleware*, the coupling occurs between software at different tiers or layers (between the middle translation layer and the data servers) whereas in the case of an FDBS, the coupling occurs between data servers that reside at the same tier. (However, this does not prevent engineers from installing into the middle tier, software for coupling between data servers.)

Quigley defines *middleware* as a software layer between the application logic and the underlying networking, security, and distributed computing technology [40]. *Middleware* provides all of the critical services for managing the execution of applications in a distributed client-server environment while hiding the details of distributed computing from the application tier [40]. Thus, *middleware* is seen in a critical role for implementing a *tightly coupled* FDBS. Similarly, Quigley considers middleware to be the key technology to integrate applications in a heterogeneous network environment [40].

## 5.6 Database integration versus database homogenization

Many organizations in both industry and government are interested in integrating autonomous, (sometimes called "stovepipe") databases into a single distributed, heterogeneous database system [8]. Many terms describe the various aspects of this integration. The multiplicity of terminology occurs because of the many ways in which databases can be integrated, and because of the many simultaneous efforts that are underway to address integration problems.

Because the degree to which *database integration* takes place will depend on the requirements and resources of the organization and its users, the term, *integration*, as it is used in various contexts has been rather vague. For people whose fields of expertise are outside the realm of database technology, software engineers have implemented techniques to hide the specific details of database-system implementation behind middleware layers and a user interface that together create the illusion of a single, unified database. By contrast, more experienced users with knowledge of multiple DBMSs can function efficiently in an environment that preserves some distinctions between the database components.

Therefore, the author proposes a comprehensive definition of *database integration* as follows. Within all architectural options, *database integration* in its broadest sense refers to the combination and transformation of database components into a database system that is homogeneous on at least one level (such as the data level, the schema level, the program interface level, or the user-interface level) and preferably, at all levels. Such an integrated database system must satisfy the primary goals of interoperability between database system components, data sharing, consistent data interpretation, and efficient data access for users and applications across multiple platforms.

Karlapalem et al. describe the concept of *database homogenization*, which they define as the process of transforming a collection of heterogeneous legacy information systems onto a homogeneous environment [33]. Whereas they do not define what they mean by the term, *homogeneous environment*, they list three goals of d*atabase homogenization*:

- to provide the capability to replace legacy component databases efficiently;
- to allow new global applications at different levels of abstraction and scale to be developed on top of the homogenized federated database; and
- to provide interoperability between heterogeneous databases so that previously isolated heterogeneous localized databases can be loosely coupled [33].

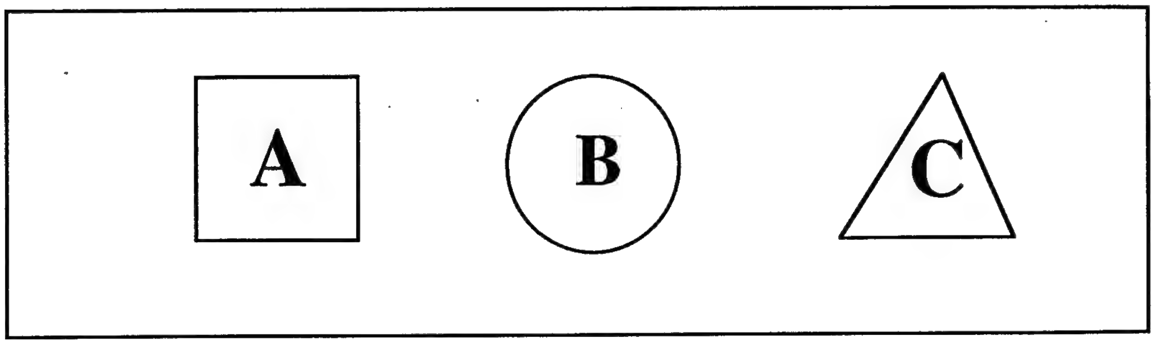
Unlike *database integration* as defined above, which explicitly includes multiple architectures and implementations, and which allows for a tightly coupled federation, the description of *database homogenization*, is associated with loose rather than tight coupling of localized databases into a homogeneous environment.

## 5.7 Data aggregation, data integration and data fusion

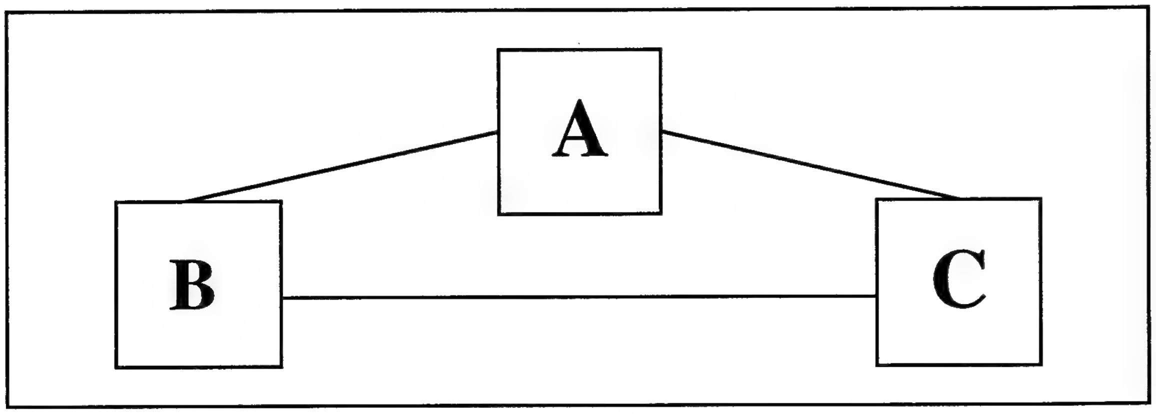
*Data aggregation* refers to the storage of or easy access to multiple data sets on the same client platform. Data aggregation, for example, occurs when a data set about submarines and another data set about electronics equipment reside in the same database, which may or may not be on the same server. (See, for example, [12]).

As described in the previous subsection, *data integration* occurs when data sets are consistent with each other and free of heterogeneity or conflicts. *Data integration* represents a tighter coupling between data sets than does *data aggregation*, where data sets are merely together, but not necessarily mutually consistent.

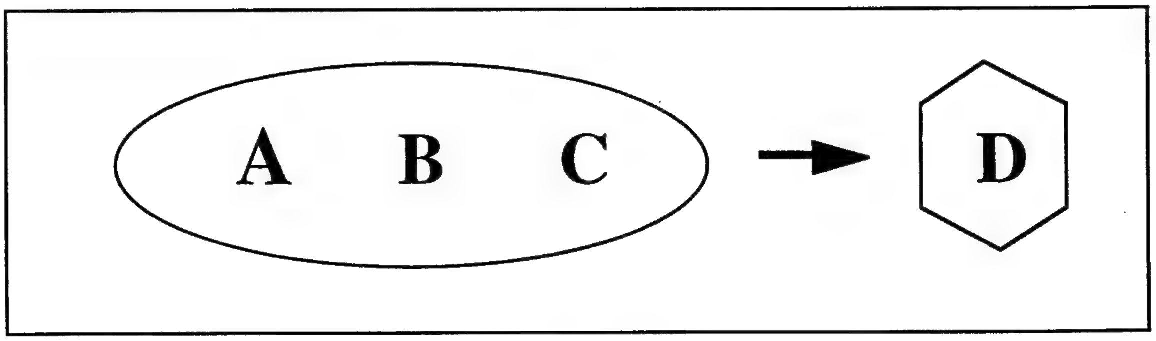
Finally, *data fusion* occurs when the information that is contained in different data sets, usually from multiple, related and complementary sources, has been analyzed, correlated and summarized into a new and unified data set that represents the consensus of the knowledge obtained in bringing together and integrating the original data sets.



a. Data Aggregation



b. Data integration



c. Data fusion

Figure 1. Conceptual comparison of data combinations with progressively tighter levels of coupling and correlation.

Figure 1 illustrates the similarities, differences and the progression between *data aggregation, data integration*, and *data fusion*. Some security implications and problems are associated with data aggregation that do not exist for data fusion because aggregated data possibly should be handled at a higher security classification than the highest classification of each individual data set in the aggregate [12]. However, because of the controlled manner in which *data fusion* is accomplished, fused data are more likely to have the appropriate classification assigned to the resulting composite data set. In *data aggregation* (Fig. 1a), data sets A, B and C are depicted with different shapes to show that they have different formats and were designed and developed using different concepts. Fig. 1b depicts a *data integration* effort that resulted in better uniformity between the three data sets, A, B and C, which are connected and now can interoperate. Fig. 1c shows the new entity, D, which is the result of a *data fusion* using data sets, A, B and C.

### 5.8 Interoperability versus interoperation

The aim of data integration is *interoperabliity*. The conditions necessary for *interoperability* are as follows [33]:

- Interconnectivity via the necessary networking facilities
- Resolution of system heterogeneity
- Resolution of semantic heterogeneity
- Derivation and/or integration of schemata and views

Whereas Karlapalem et al. do not elaborate further on the meaning of *system heterogeneity*, Ceruti and Kamel have described three levels of heterogeneity including platform heterogeneity, data model heterogeneity, and semantic heterogeneity [7]. Excluding semantic heterogeneity, the term *system heterogeneity* is seen to be some combination of platform heterogeneity (different DBMS software and implementation) and data model heterogeneity (schemata, query languages, integrity constraints and nullness requirements.) Since Karlapalem et al. already have listed the integration of schemata as an item separate from system heterogeneity, *system heterogeneity* logically should refer to the differences between DBMS vendors, transaction processing algorithms, query languages, query optimization techniques, integrity constraints, and nullness requirements. If this definition is assumed for *system heterogeneity,* the necessary conditions for database interoperability listed above become sufficient conditions.

Similarly, Drew et al. have correctly stated that computer system-heterogeneity and data-management-system heterogeneity must be resolved as a requirement for interoperability among existing information systems [22].

That not withstanding, the achievement of database interoperability simply supplies users and applications with only the ability to interoperate in a common-data environment. It does not guarantee that *interoperation* will occur. *Database interoperation* results when users and applications take advantage of a common, integrated environment to access, share, and process data across multiple databases. This concept also extends to interoperation among knowledge bases [10].

### 5.9 Legacy information system versus migration information system

Autonomous systems that become candidates for integration into a global, distributed system sometimes have been called *migration systems*. These systems are supported by *migration information system* with *migration databases*. The term *migration databases* indicates unambiguously that the database in question has been chosen to be included in some form of a global database system, particularly a distributed system, such as an FDBS. By contrast, the term *legacy information system* has been used in two different ways. At one extreme, some use *legacy information system* and *legacy database* to be synonymous with *migration information system* and *migration database*, respectively.

Others have referred to a *legacy information system* as though it were obsolete and not a *migration information system* and, therefore, deliberately excluded from the final integrated database configuration. This is the opposite extreme. More common than the extreme cases, a subset of legacy data is deemed important to the users of a shared data resource. This means that some or all of the data in a *legacy* *information system* may be migrated during a database-integration effort. For example, Cykana describes steps in the data-integration process that start with the movement and improvement of data, and progress to the shutdown of legacy systems [15]. Karlapalem et al. refer to the difficulty of migrating *legacy information systems* to a modern computer environment [33] in which some difference is presumed to exist between the legacy system and the modern system.

The author recommends that the following terminology be adopted as standard: *Legacy data* and *legacy information system* should refer to the original data and system in their original formats, as maintained in the original, autonomous information system before any modification or migration to a new environment has occurred. *Migration data* and *migration information system* should be used to describe the subset of the legacy data and software that has been chosen to be included in a new (and usually distributed) information-resource environment. When data and software are modified to accommodate a new environment, they should be called *migration* instead of *legacy*.

## 6. Terms associated with semantic heterogeneity

Semantic heterogeneity refers to a disagreement about the meaning, interpretation or intended use of the same or related data or objects [43]. Semantic heterogeneity can occur either in a single DBS, in a multidatabase system or in a knowledge base system. Its presence in a DBS also is independent of data model or DBMS. Therefore, the terminology associated with this problem is discussed in a separate section.

### 6.1 Semantic interoperability versus database harmonization

The terms, *database integration* and *interoperability* were discussed above in a general context. For distributed, heterogeneous database systems to be integrated in every respect, semantic heterogeneity must be resolved [7]. Because problems associated with semantic heterogeneity have been difficult to overcome, they have received considerable attention in the literature [6, 7, 22, 41 and 43]. The terminology to describe semantic heterogeneity also has evolved. For example, Sciore et al. define *semantic interoperability* as agreement among separately developed systems about the meaning of their exchanged data [41].

Whereas the exact meaning of the term, *database harmonization* is not clear, one can infer that the goal of *database harmonization* must be related to providing an environment in which conflicts have been resolved between data representations from previously autonomous systems. This further implies that the resolution of semantic heterogeneity is a prerequisite for *database harmonization*. Although a more precise definition of *database harmonization* needed, it appears to be related to the idea of *semantic interoperability* and database integration (See subsection 5.6)

### 6.2 Strong and weak synonyms versus class-one and class-two synonyms

A *synonym* is a word that has the same or nearly the same meaning as another word of the same language [48]. Because a metadata representation will include more attributes (data element name, type, length, range, and domain) than ordinary nouns, it was necessary to consider various levels of similarity, and therefore, levels of synonymy.

Bright et al. described the concept of *strong* and *weak synonyms* [6]. *Strong synonyms* are semantically equivalent to each other and can be used interchangably in all contexts without a change of meaning, whereas *weak synonyms* are semantically similar and can be substituted for each other in some contexts with only minimal meaning changes [6]. It follows from the definition of *weak synonyms* that they cannot be used interchangably in all contexts without a major change in the meaning, a change that could violate the schema specification.

This concept is similar to one that was introduced in [7] in which two classes of synonym abstraction were defined, *class one* and *class two*. *Class-one synonyms* occur when different attribute names represent the same, unique real-world entity [7]. The only differences between class-one synonyms are the attribute name and possibly the wording of the definition, but not the meaning [7]. By contrast, *class-two* *synonyms* occur when different attribute names have equivalent definitions but are expressed with different data types and/or data-element lengths [7]. Class-two synonyms can share the same domain or they can have related domains with a one-to-one mapping between data elements, provided they both refer to the same unique real-world entity [7].

The concept of a *strong synonym* is actually the same as that of a *class-one synonym,* because both *strong synonyms* and *class-one synonyms* are semantically equivalent and they can be used interchangeably because they have the same data element type and length. By contrast, the concept of a *class-two synonym* includes, but is not limited to, the concept of a *weak synonym* because the definition of a *weak synonym* seems to imply a two-way interchange in some contexts. The main difference is that the possible interchangability of class-two synonyms is determined, not only by semantic context, but also by the intersection of their respective domains, as well as their data types and lengths. *Class-two synonyms* allow for a one-way, as well as a two-way, interchange in some cases. By contrast, the "each-other" part in the definition of *weak synonyms* seems to focus mainly on the two-way interchange and may preclude a situation in which only a one-way interchange is possible. For example, a shorter character string can fit into a longer field, but not vice versa.

## 7. Summary

This paper presents a review of the rapidly growing vocabulary of database system technology, along with its conflicts and ambiguities. Solutions are offered to address some of the problems encountered in communicating concepts and ideas in this field. This effort is intended to be a step toward the development of a more comprehensive, standard set of terms that can be used throughout the industry. More work is needed to identify and resolve the differences in interpretation between the many terms used in information technology as they occur in industry, government, and academia.

## Acknowledgments

This work was produced by a U.S. government employee in the course of employment and is not subject to copyright. It is approved for public release with an unlimited distribution.

## References

[1] American National Standards Institute, Inc., *American National Standard for Information Systems - Information Resource Dictionary System (IRDS),* ANSIX3.138-1988, Federal Information Processing Standard (FIPS) Pub 156, pp. 1-7 to 1-16, New York, 1988.

[2] American National Standards Institute, Inc., *Ibid.*, pp. 1-42 to 1-46.

[3] K. S. Brathwaite, *Relational Databases - Concepts Design and Administration,* McGraw-Hill, New York, NY., pp. 6-7 (1991).

[4] K. S. Brathwaite, *Ibid.*, pp. 61-87.

[5] Y. Breitbart, and A. Silberschatz, "Multidatabase Update Issues," in *Proceedings of the ACM SIGMOD Conference*, pp. 135-142, June 1988.

[6] M. W. Bright, A. R. Hurson and S. Pakzad, "Automated Resolution of Semantic Heterogeneity in Multidatabases," *ACM Transactions on Database Systems*, vol. 19, no. 2, pp. 212-253, June 1994.

[7] M. G. Ceruti and M. N. Kamel, "Semantic Heterogeneity in Database and Data Dictionary Integration for Command and Control Systems," in *Proceedings of the 11th Annual DoD Database Colloquium '94,* pp. 65-89, Aug. 1994.

[8] M. G. Ceruti, M. N. Kamel, and B. M. Thuraisingham "Object-Oriented Technology for Integrating Distributed Heterogeneous Database Systems," in *Proceedings of the 12th Annual DoD Database Colloquium '94,* pp. 79-98, Aug. 1995.

[9] M. G. Ceruti, "A Review of Data Base System Terminology," *Handbook of Data Management 1996-97 Yearbook*, Chap. I-1, pp. S3-S17, B. Thuraisingham, editor, Auerbach Publications, Boston, 1996.

[10] M. G. Ceruti, "Application of Knowledge-Base Technology for Problem Solving in Information-Systems Integration," *Proceedings of the DoD Database Colloquium '97,* pp. 215 -234, Sep. 1997.

[11] M. G. Ceruti, "A Review of Data Base System Terminology," *Handbook of Data Management 1998,* Chap. 1, pp. 3-21, B. Thuraisingham, editor, Auerbach Publications, CRC Press LLC, Boca Raton, 1998.

[12] M. G. Ceruti, "Challenges in Data Management for the United States Department of Defense (DoD) Command, Control Communications, Computers and Intelligence ($C^4I$) Systems," *Proceedings of the Twenty-Second Annual IEEE International Computer Software and Applications Conference, IEEE COMPSAC'98*, pp. 622-629, Aug. 1998.

[13] M. G. Ceruti and M.N. Kamel, "Heuristics-Based Algorithm for Identifying and Resolving Semantic Heterogeneity in Command and Control Federated Database Systems," *Proceedings of IEEE Knowledge and Data Engineering Exchange Workshop, KDEX'98*. pp. 17-26, Nov. 1998.

[14] M. G. Ceruti, "Web-to-Information-Base Access Solutions," in *Handbook of Local Area Networks 1999,* pp. 485-499, J. P. Slone, editor, Auerbach Publications, CRC Press LLC, Boca Raton, 1999.

[15] P. Cykana, "Defense Information Infrastructure: Data Migration Tasks, Techniques, and Solutions," *Proceedings of the DoD Database Colloquium '95,* pp. 3-13, Aug. 1995.

[16] C. J. Date, *Relational Database Selected Writings*, p. 487-490, Addison-Wesley, Reading, MA, 1986.

[17] C. J. Date, *Relational Database Writings 1985-1989*, p. 271, Addison-Wesley, Reading, MA, 1990.

[18] C. J. Date, "Domains, Relations and Data Types," *Database Programming and Design,* vol. 7 no. 6, pp. 19-21, June 1994.

[19] H. Darwen and C. J. Date, "Introducing the Third Manifesto," *Database Programming and Design*, vol. 8, no. 1, pp. 2 -35, Jan. 1995.

[20] H. Darwen and C. J. Date, "The Third Manifesto," *SIGMOD Record*, vol. 24, no. 1, pp. 39-49, Mar. 1995.

[21] Defense Information Systems Agency, *Command and Control (C2) Core Data Model*, Sep. 1993.

[22] P. Drew, R. King, D. McLeod, M. Rusinkiewicz and A. Silberschatz, "Report of the Workshop on Semantic Heterogeneity and Interoperation in Multidatabase Systems," *SIGMOD Record*, vol. 22, no. 3, pp. 47-56, Sep. 1993.

[23] S. R. Finlow, "An Analysis of Date's Twelve Rules for Distributed Database Systems," *Proceedings of the DoD Database Colloquium '94*, pp. 581-591, Aug. 1994.

[24] P. J. Fortier, D. Fisher, D. K. Hughes, and M. Roark, "Final Report of the DBSSG Predictable Real-Time Information System Task Group," *Proceedings of the DoD Database Colloquium '95,* pp. 185-195, Aug. 1995.

[25] P. Goodyear, H. W. Ryan, S. R. Sargent, S. J. Taylor, T. M. Boudreau, Y. S. Arvanitis, R. A. Chang, J. K. Kaltenmark, N. K. Mullen, S. L. Dove, M. C. Davis, J. C. Clark and C. Mindrum, *Netcentric and Client/Server Computing: A Practical Guide*, Chapter 29, pp. 29-1 to 29-17 Andersen Consulting, Auerbach Publications, CRC Press LLC, Boca Raton, 1999.

[26] P. Goodyear, et al., *Ibid,* Chapter 31, pp. 31-1 to 31-24.

[27] J. Glymph, "Data Standardization in the US Army," *Proceedings of the DoD Database Colloquium '91*, pp. 1-36, June 1992.

[28] M. Hammer and D. McLeod, " *On Database Management System Architecture* ," Tech. Rep. MIT/LCS/TM-141, Massachusetts Institute of Technology, Cambridge, MA, 1979.

[29] M. Hammer and D. McLeod, "On Database Management System Architecture," Tech. in *Infotech State of the Art Report* vol. 8: Data Design, Pergamon Infotech Limited, 1980.

[30] C. Imhoff and B. Birrus-Montanari, *Designing the Operational Data Store,* Barnett Data Systems, Rockville, MD., Intelligent Solutions, Inc., 1997.

[31] W. H. Inmon, *Building the Data Warehouse*, Barnett Data Systems, Rockville, MD.

[32] W. H. Inmon C. Imhoff, and G. Battas, *Building the Operational Data Store,"* 1997 Intelligent Solutions, Inc.

[33] K. Karlapalem, Q. Li, and K.D. Shum, "HOFDA: An Architectural Framework for Homogenizing Heterogeneous Legacy Databases," *SIGMOD Record*, vol. 24, no. 1, pp. 15-20, Mar. 1995.

[34] W. Kent, "A Simple Guide to Five Normal Forms in Relational Database Theory," *Communications of the ACM*, vol. 26, no. 2, pp. 120-125, Feb. 1983.

[35] A. King, M. Koltz, T. Jones, and K. Stanford, "Repository Implementation in a Legacy/Reengineering Environment," *Proceedings of the DoD Database Colloquium '95,* pp. 585-592, Aug. 1995.

[36] B. S. Lee, "Normalization in OODB Design," *SIGMOD Record*, vol. 24, no. 2, pp. 23-27, Sep. 1995.

[37] H. J. Levesque and R. J. Brachman, "A Fundamental Tradeoff in Knowledge Representation and Reasoning (Revised Version)," *The Knowledge Representation Enterprise*, Chap. 4, pp. 41-70; Original version appeared as H. J. Levesque, "A Fundamental Tradeoff in Knowledge Representation and Reasoning," *Proceedings of CSCSI/SCEIO Conference (CSCSI-84)*, London, Ontario, pp. 141-152, May 1984.

[38] J. Little, "Using Expert System Technology to Standardize Data Elements," *Proceedings of the DoD Database Colloquium '95*, pp. 205-217, Aug. 1995.

[39] W. Litwin, J. Boudenant, C. Esculier, A. Ferrier, A. Glorieux, J. La Chimia, K. Kabbaj, C. Moulinoux, P. Rolin, and C. Stangret, "SIRUS Systems for Distributed Data Management," *Distributed Data Bases*, H. -J. Schneider, editor, North-Holland, Netherlands, pp. 311-366, 1982.

[40] G. V. Quigley, "Message-Oriented Middleware (MOM): A Key Technology for the Successful Deployment of Distributed Client/Server Information Systems," *Proceedings of the DoD Database Colloquium '95*, pp. 297-309, Aug. 1995.

[41] R. Sciore, M. Siegel, and A. Rosenthal, "Using Semantic Values to Facilitate Interoperability Among Heterogeneous Information Systems," *ACM Transactions on Database Systems*, vol. 19, no. 2, pp. 254-290, June 1994.

[42] H. Simon, "Biotechnology Data Warehousing," *Today's Chemist at Work*, vol. 7, no. 10, pp. 183 - 236, Nov. 1999.

[43] A. P. Sheth, "Semantic Issues Multidatabase Systems," *SIGMOD Record*, vol. 20, no. 4, pp. 5-9, Dec. 1991.

[44] A. P. Sheth and J. A. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases," *ACM Computing Surveys*, vol. 22, no. 3, pp. 183 - 236, Sep. 1990.

[45] B. M. Thuraisingham, "Data Warehousing and Data Mining: Developments and Challenges," *Proceedings of the DoD Database Colloquium '96*, pp. 79-80, Aug. 1996.

[46] B. M. Thuraisingham, *Data Mining: Technologies, Techniques, Tools, and Trends*, CRC Press LLC, Boca Raton, 1998.

[47] L. S. Waldron, "Natural Language Generator Based on Database Modeling," *Proceedings of the DoD Database Colloquium '95*, pp. 251-295, Aug. 1995.

[48] N. Webster, *Webster's New Universal Unabridged Dictionary*, Deluxe Second Edition, (Webster's New Twentieth Century Dictionary), J. L. McKechnie, editor, Simon and Schuster, New York, N. Y., 1983.

[49] L. Wheeler, "From Utter Simplicity to Chaotic Complexity (and Back Again): A Conceptual System Structure for Data Administrators," *Proceedings of the DoD Database Colloquium ' 94*, pp. 19-37, Aug. 1994.

[50] P. Winsberg, *Sybase Relational Database Design Student Guide*, version 4, Sybase Corp., 1990.

[51] M. K. Wysong, "Metadata: Key to Successful Data Warehouse Projects," *Proceedings of the DoD Database Colloquium '96*, pp. 125-132, Aug. 1996.

**Dr. Marion G. Ceruti** is a scientist in the Advanced Concepts and Engineering Division of the Command and Control Department at the Space and Naval Warfare Systems Center, San Diego. She received the Ph.D. in 1979 from the University of California at Los Angeles. Dr. Ceruti's present professional activities include information systems research and analysis for command and control decision-support systems. She has served on the program committee and as Government Point of Contact for Database Colloquia since 1987. An active member of AFCEA, IEEE and several other scientific and professional organizations, Dr. Ceruti is the author of numerous publications on various topics in science and engineering.

# REPORT DOCUMENTATION PAGE

*Form Approved OMB No. 0704-0188*

| | | |
|---|---|---|
| 1. AGENCY USE ONLY *(Leave blank)* | 2. REPORT DATE<br>September 1999 | 3. REPORT TYPE AND DATES COVERED<br>Professional Paper |

| | |
|---|---|
| 4. TITLE AND SUBTITLE<br>An Expanded Review of Information-System Terminology | 5. FUNDING NUMBERS<br>In-house |
| 6. AUTHOR(S)<br>M. G. Ceruti, Ph.D. | |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)<br>Space and Naval Warfare Systems Center<br>San Diego, CA 92152–5001 | 8. PERFORMING ORGANIZATION REPORT NUMBER |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)<br>Space and Naval Warfare Systems Center<br>San Diego, CA 92152-5001 | 10. SPONSORING/MONITORING AGENCY REPORT NUMBER |
| 11. SUPPLEMENTARY NOTES | |
| 12a. DISTRIBUTION/AVAILABILITY STATEMENTR<br>Approved for public release; distribution is unlimited. | 12b. DISTRIBUTION CODE |

13. ABSTRACT *(Maximum 200 words)*

Terminology pertaining to database systems is reviewed, particularly with respect to relational database systems: heterogeneous, distributed database systems; and information-management methods. The literature of a variety of database researchers and data administrators is included in this review.

A comparison between the different ways in which some of the terminology is used in the industry is presented. In some cases, different definitions of the same term can be consistent when these definitions pertain to different aspects of the entity that the term represents. In other cases, popular misconceptions about word meanings are explained. Resolutions to conflicts in meaning and usage are suggested.

Both the similarity and the diversity of ideas concerning the most basic, as well as the more complex database concepts are covered. For example, the discussions range from an examination of the word, data, in the general section, to the classification of synonyms in the section on semantic heterogeneity. This expanded review contains further comparison between types of data associations and between information storage and retrieval methods. It includes a comparison between knowledge bases and databases.

Published in Proceedings of AFCEA's Sixteenth Annual Federal Database Colloquium and Exposition, "Information Dominance and Assurance," September 21-23, 1999.

| | |
|---|---|
| 14. SUBJECT TERMS<br>Mission Area: Command and Control<br>database system terminology<br>information system<br>nomenclature | 15. NUMBER OF PAGES |
| | 16. PRICE CODE |

| | | | |
|---|---|---|---|
| 17. SECURITY CLASSIFICATION OF REPORT<br>UNCLASSIFIED | 18. SECURITY CLASSIFICATION OF THIS PAGE<br>UNCLASSIFIED | 19. SECURITY CLASSIFICATION OF ABSTRACT<br>UNCLASSIFIED | 20. LIMITATION OF ABSTRACT<br>Same as Report |